FACILITY FORM 602

N65-17951
(ACCESSION NUMBER)

(THRU)

87
(PAGES)

1
(CODE)

CR-60925
(NASA CR OR TMX OR AD NUMBER)

08
(CATEGORY)

STRUCTURAL ANALYSIS SYSTEM
USAGE REPORT

ROBERT J. MELOSH, PHILIP A. DIETHER, MARY BRENNAN, AUTHORS

GPO PRICE $ ____________

OTS PRICE(S) $ ____________

Hard copy (HC) $3.00

Microfiche (MF) $0.75

PHILCO

A SUBSIDIARY OF
Ford Motor Company

WESTERN DEVELOPMENT LABORATORIES

3875 FABIAN WAY • PALO ALTO, CALIFORNIA

#63-296

WDL-EM0763
July 1963

STRUCTURAL ANALYSIS SYSTEM
Usage Report

by

Robert J. Melosh, Philip A. Diether, Mary Brennan

Prepared by

PHILCO CORPORATION
A Subsidiary of the Ford Motor Company
Western Development Laboratories
Palo Alto, California

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration under Contract NAS7-100.

Contract No. 950321

Prepared for

JET PROPULSION LABORATORY
California Institute of Technology
Pasadena, California

ABSTRACT

14951

This report describes the computer aspects of the Structural Analysis System. It includes a general description of system components and their function, operational, flow, and program intelligence. It describes data and program identification, formats, and handling. It defines system tape and core assignments. It includes writeups of each of the subprograms and discusses requirements for preparing input data.

The theoretical basis for the program is contained in a companion report entitled "Structural Analysis System - Technical Report."

FOREWARD

The work reflected in this report was performed by the Philco Corporation, Western Development Laboratories, Palo Alto, California, for the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, under Contract No. 950321. This work is part of a continuing effort to develop automated capability to solve problems in engineering mechanics. Ted [redacted] Lang and Robert Reed of the Jet Propulsion Laboratory were the Project Engineers. The research was conducted from February through July 1963 by the Engineering Mechanics Section of WDL.

Phil R. Cobb, Manager, Electro-Mechanical Department, Philco, was Project Manager for the contractor. Robert J. Melosh, Manager, Engineering Mechanics, was Project Engineer.

The principal engineers for the contractor were Henry Christiansen and Philip Diether. Mary Brennan was the senior programmer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INDEX OF USAGE WRITEUPS

1. Introduction

The high speed digital computer has been an increasingly popular tool for structural analysis. A multitude of programs has been generated to handle small-deflection behavior prediction for specialized classes of structures such as trusses, frames, and shells of symmetric forms and loading. Many of these programs are highly efficient packages in terms of economically producing the desired results. On the other hand, because they are specialized, the analyst must become familiar with a vast array of them, or write his own program, unless the scope of his work is parochial. The one or two programs which exist which are sufficiently general have limited flexibility and treat problems of limited size. The first restriction is important because it limits the use of the program and complicates data handling. The second restriction is significant because the persistent trend in computer application is to continually increase problem sizes because cost per calculations are decreasing and better analyses become justifiable.

The Structural Analysis System is being developed to simplify automated structural analysis and to eliminate reprogramming for problem changes. These two objectives are, to some extent, in conflict. They are reconciled by choosing the following programming concepts:

1. Wherever possible, standardize. This is achieved in the program for output formats, most input formats, error handling, tape handling and formats, and storage assignment.
2. Provide a modular program. This is achieved by dividing the calculation into a number of tasks which can be performed in the sequence specified by the analyst. A single intelligence system is used. This approach facilitates adding or removing modules from the system.

3. Program for intermediate size problems. This is achieved by providing for the use of tapes for data and program storage. To make this operation efficient, tape search is avoided wherever possible.

This report describes in general and in detail how calculations are directed and how data must be prepared for the Structural Analysis System (SAS). Material is organized in six sections. In the first section a general description of the system, its components and flow are included. The second section discusses the program control. The third section describes the pseudo-instructions. The fourth section describes the operation subprograms, their data formats and usage. The fifth section describes the BILD operation subroutines which, because of their number and size are included in a separate section. The final section discusses input data.

2. Structural Analysis System Characteristics

In this section are described the scope of the program, general information, and the basic concepts on which the program is based. Detailed description of program components are included in other sections.

2.0 General Aspects

The primary objective of this program is to automate linear structural analysis of shells. This includes prediction of deflections and stresses of structures under pressure, heat, inertial and static loads. In addition resonant frequencies can be obtained. Materials must have linear stress-strain relations, deflections must be such that load-deflection relations are linear.

However, the program is also suitable for problems involving the same or similar partial differential equations or problems involving matrix algebra. Such problems would be encountered in electrical networks, potential fields, fluid flow, heat conduction and Dirichlet and Newman equations in physics.

This program can handle matrices of intermediate size (500-10,000) efficiently as well as smaller sizes (less than 500) with a small penalty as compared to a specialized in-core routine. Whenever possible problems are kept in core to avoid the use of the tapes which are inherently slower. Answers are obtained in a single pass on the machine. Emphasis has been placed on self-checking, simple recovery from faults, and preservation of calculations up to the fault point. Application flexibility is attained by leaving operation sequence and data disposition under control of the analyst.

2.1 System Components

The system is composed of four components: The Master Program Control, the pseudo instructions, the operation subprograms, the input data. The Master Program Control (MPC) includes the master intelligence and the operator subroutines. MPC remains in core during the execution of all pseudo instructions. MPC subroutines control tape search, tape reading, matrix coding, and error operations. The pseudo instructions define data assignments, data tape assignments, matrix identification and the sequence of subprogram selection. The operation subprograms perform operations on matrices of data. Matrix sorting, addition, inversion, multiplication, input and output handling are some examples of operations performed. Input data are required by the subprograms to define the particular problem under consideration.

2.2 System Flow

Initially MPC transfers control to the subprogram for reading a set of pseudo instructions and generating a pseudo program (MAKER) from the pseudo instructions. The pseudo program tape is written and rewound, then the pseudo instructions are read into core one at a time. For each instruction, MPC locates the required data, assigns core storage areas, reads the subroutine if required, and transfers control to the subroutine. Upon return from the subroutine, MPC provides error diagnostics and then reads and executes the next pseudo-instruction.

As an example, the steps taken by the computer are defined in Table 2-1 for generating two component stiffness matrices and summing them. This problem includes all the steps characteristic of a more complex problem involving other subprograms and subroutines. All steps are independent of matrix size.

TABLE 2-1

EXAMPLE PROBLEM

Step Number	Operation
1	Read the Master Program Control from tape
2	Read the pseudo instructions and generate (MAKER) the program
3	Read a generated pseudo instruction
4	Select the BILD program from tape and read into core
5	Transfer control to the BILD package
6	Generate matrices and write on tape
7	Return Control to the Master Program Control
8	Read the addition (ADDS) pseudo instruction
9	Locate the appropriate matrices from core or tape
10	Select the subprogram and read from tape.
11	Enter the subprogram (ADDS) and add the matrices
12	Read the HALT pseudo-instruction
13	Halt

2.3 Core and Tape Assignments

Core storage is divided into three basic regions: systems, program, and common. Systems storage contains the IBSYS, MONITOR or other computer job system components necessary for job control.

Programs in storage include the MPC program and the current subprogram being used for a particular operation. MPC always remains in core during the solution of a given problem. Subprograms are brought into core as they are needed.

Common storage contains the parameters common to most of the subprograms. They are the following one-dimensional arrays.

TABLE 2-2.1 COMMON DATA

	Array	Dimension	Location	Function
1.	ABA	20	1	Logic Pseudo Code Table
2.	ABB	100	21	Operation Code and Subprogram Designation Table
3.	BUR	12	121	Pseudo Instruction
4.	NCE	1	133	Word Error Indicator
5.	TRA	30	134	A, B, and C Matrix Identification Data
6.	KØD	5	164	Word Region for Forming and Decomposing Codes
7.	NLD	1	169	Word Defining the Dimension of DAT
8.	NBD	1	170	Word Defining the Number of Blocks in DAT
9.	REE	120	171	Storage Region for Reading Tape Blocks
10.	WRY	120	291	Storage Region for Writing Tape Blocks
11.	DAT	x	411	Data Storage Region for Matrices and Erasable Storage

The ABA table defines all acceptable logic pseudo instructions. The ABB table defines all acceptable operation instructions and the designations of the respective operation subprograms. All pseudo instructions are checked against the instructions in these tables.

The pseudo instructions region (BUR) contains a current instruction being operated on by the MPC. The identification region (TRA) defines the A, B, and C matrices being operated with, or, the most recent ones operated upon. The KOD region is used by COINS in the process of coining or decomposing row and column codes. The function of the read (REE) and write (WRY) blocks is to provide a space where data can be stored when it is read to and from tape. These blocks serve as buffers in core for the tape. For further explanation of these regions see the "common" operation subprogram writeup entitled NAME.

The data region (DAT) is the largest array and encompasses the remaining common storage in core. This region is broken into one, two, or three parts depending upon how many matrices are being handled and the storage assigned by MPC.

Table 2-3 gives a list of the tape numbers and their assignments. Whenever possible the use of the scratch tapes 7 and 8 should be avoided for intermediate calculations since these tapes are used by some subprograms for intermediate data storage. This means that data on these tapes may be erased by a subprogram. The use of matrix tapes should be avoided if possible. The primary reason is because tape usage is a slow process compared to in-core operation. Core operation should always be used if the matrices will fit in core. However, if tapes must be used because of the size of the problem then as many tapes as possible should be used to minimize tape search time.

Writing and reading of matrix tapes is performed only by the TAPES subroutine. If the use of storage disks rather than tapes is used at some future time this routine only will need to be changed. Data are assigned to matrix tapes in conformance with the pseudo instructions (Section 4).

2.4 Data Handling

The handling of data can be divided into three categories: the handling of input, in-transit, and output data. Input data is always read from Tape 5. The input data can be of the following kinds: pseudo instructions, material tables, element data, matrices and matrix heading cards. Pseudo instructions are always the first data read in for a job. After the pseudo program has been generated it calls in the next three kinds of data as they are needed.

In-transit data includes matrices and subprograms. Matrices can be sent to and from scratch tapes or matrix tapes. The form of matrices sent to or received from scratch tapes depends on the subprogram writing the tapes when these tapes are used as intermediate storage in the subprogram. However, matrices sent to or received from matrix tapes will be in one of two forms; either coded or pre-coded. Scratch tapes will be written in the same way as matrix tapes if they are designated as output matrix tapes in a pseudo instruction.

In coded format the matrix consists of a one-dimensional array of two alternating terms. The odd numbered terms are called codes. They are packed words defining the row and column numbers of a matrix coefficient. The even numbered terms are the elements. They are the value of the matrix element at the location defined by the preceding code. If a zero code occurs, this indicates that the array is terminated. It is not necessary that a coded matrix be sorted.

TABLE 2 - 3

LOGICAL TAPE ASSIGNMENTS

(Load Time)

Tape No.	Actual Unit	SAS Assignment	JPL Assignment
1	A1	Systems Tape	IBSYS
2	B2	Optional Systems Tape	Unassigned
3	B3	Pseudo Program Tape	Unassigned
4	A4	Chain Tape (SAS Library)	Chain Tape
5	A2	Input Data Tape	Input Data Tape
6	A3	Printer Output Tape	Printer Output Tape
7	B4	Scratch Tape	Punch Output Tape
8	B1	Scratch Tape	Unassigned
9		Matrix Tape	Plotter Tape
10		Matrix Tape	Plotter Tape
11		Matrix Tape	Systems Library
12		Matrix Tape	Unassigned
13		Optional Matrix Tape	Unassigned
14		Optional Matrix Tape	Unassigned
15		Optional Matrix Tape	Unassigned
16		Optional Matrix Tape	Unassigned

In precoded format the matrix is a one-dimensional array consisting of three parts. The parts are the row codes, column codes, and elements. The row code is a packed word containing the row node number and component number. The column code is similar. The elements are the values of the matrix. The order of the three parts depends upon whether the matrix is row listed or column listed. If row listed, the column codes are listed first immediately followed by the row codes. Then, beginning a new block, the matrix elements are listed by rows. If column listed, the row codes, then column codes, and finally elements are listed, the elements being listed by columns.

Subprograms are in-transit data controlled by the chain subroutine which is a part of the job system. Chain searches tape and brings subprograms from tape to core. Since subprograms are always read into the program storage area, previous subprograms in core are destroyed when the new subprogram is read in.

Output data is in three forms: matrices, error statements and error exit data. Matrices are printed in either coded or precoded format by INKS. "Heading" statements desired by the analyst are printed out with the A matrix only. Heading cards are read from the input tape and immediately printed one card at a time. Heading statements are always listed on separate pages from the listing of the matrix. Their format is A72.

Error statements are those error comments printed out by a subprogram. Error comments are printed when an error is encountered. Insofar as possible all format, listing, identification, and xize checks are performed before the error exit is taken from a subprogram.

Error exit data output occurs when a nonrecoverable or flow disrupt error occurs. In these cases the status of the core and tapes is identified by printing the matrix identification data along with the instruction being worked on. Matrix identification includes all pertinent information of the TRA block (See Section 5). The error exit printout is controlled by the subprogram ERROR.

Data and identification information are stored on matrix tapes in blocks of 127 words each. The first 120 words are data and the last seven words are matrix and tape identification information (TRA 1-8). The first word is the matrix alphabetic identification; the second word is the matrix numeric identification; the third, the number of the block of data; the fourth the number of rows and columns in the matrix; the fifth, the listing of the data, row, column, or unsorted; the sixth the format of the coding, coded or preceded; and the seventh, the tape-group number on the tape. The purpose of including the last piece of information is to provide the MPC with the capability of backspacing records on tape and of checking to make certain that the tape record is the one required and in proper sequence.

Group numbers, which identify the location of matrices on tape, must be in sequential order, e.g., 1, 2, 3, 4.... Also, it is not permissible to write a new matrix in the middle of a sequential group. The new matrix may be shorter or longer than the one it is being written over, in which case, the identification associated with the overwritten or succeeding matrix will be partially destroyed, thus interfering with subsequent logical tape search.

Tape search is accomplished as follows. The tape backspaces oné block, checks the group number, then determines whether to move forward or backward to reach the correct group number. If, after the one backspace, the block read is in the group desired the tape looks at the block number and backs up the absolute value of the number. This positions the tape at the beginning of the group to be read.

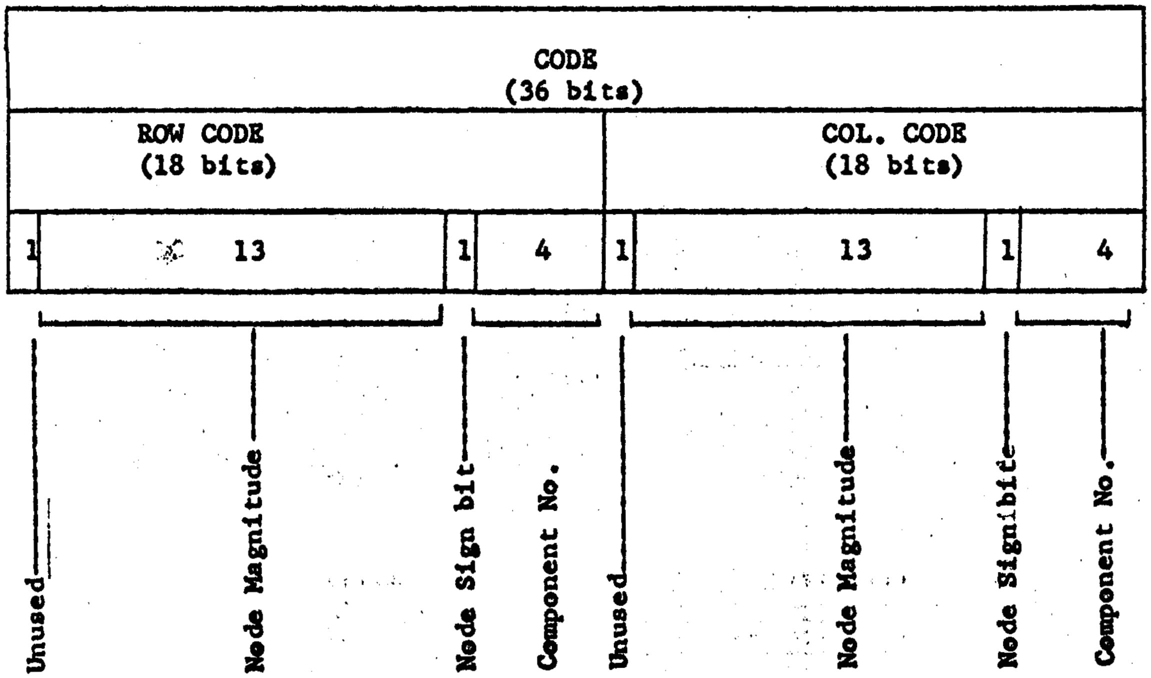
2.5 Identification Systems

Three systems are used to discriminate between subprograms, matrices, and data. Subprograms are identified by a chain number on the program tape. The correspondence between chain numbers and subprogram instructions is contained in COMMON in the ABB table. Further information on chain numbers is given in Section 2.6.

Matrices are identified by two words. The first word consists of three alphabetic characters. The second word consists of three numeric characters, e.g. MAT106. Other identification which characterizes the form of a matrix is stored in the TRA region of common storage and in each block of data on tape.

The data in the matrices are identified by codes. The codes are 17 bit packed words containing column, row, and component data identifying the nodes of the structure. The coding system is based on the nodal numbers associated with the matrix during its development and the directional component at that node. A maximum of 4095 nodes are admissible by the coding system although it is possible to solve larger problems in pieces. A maximum of eight components of force or displacement may be considered at a node.

A graphical breakdown of the codes appears in Fig. 2-5. Three bits of the 17 bit word are reserved to define the component at the node and 13 bits to define the nodal number. The remaining bit designates the coordinate system used. When only the row or column number is required as in the precoded matrices the 17 bits are put in their upper part of the word to identify the row or column.



CODE WORD BREAKDOWN

Fig. 2.5

2.6 Subprogram Handling

Subprograms are handled using the IBM CHAIN subroutine. During preload time, the subprograms are stored on the SAS program tape. An initiating subroutine calls CHAIN and directs read-in of the MPC. The first subroutine in MPC is SPACE. This subroutine is stored at the origin and reserves the space required for the largest operation subprogram.

Upon call for an operation subprogram, the chain table (ABB) is searched to find the chain number of the instruction, and control is transferred, if the subprogram is not already in core, to CHAIN. This subroutine searches the SAS tape, reads the required subprogram, and transfers control to it. The exit instruction taken from the subprogram is a call to the CYCLE subroutine. This is a FAP subroutine which transfers control back to MPC.

Subprogram chain numbers are given in Table 2-4. This table also includes the subprogram identification used to label source and binary decks.

TABLE 2-4

SUBPROGRAM IDENTIFICATION

Program	Chain Number	Label
READ	1	A
MAKER	2	B
COLS	3	C
ROWS	4	D
FLIP	5	E
CODE	6	F
DECO	7	G
ADDS	8	H
MULT	9	I
WASH	10	J
ITER	11	K
CHIN	12	L
CHOL	13	M
SORT	14	N
ROOT	15	O
BILD	16	P
SUBS	17	Q
INKS	18	R
TAPES		W
COINS		X
MINTS	26	Z

If a format error is encountered in the subprogram, control is transferred to the subroutine EXEM. This subroutine in turn sets up the required error indicator, points diagnostic comments, checks remaining input if reasonable, and transfers to CYCLE.

Two special subroutines are used to handle two particular types of non-recoverable errors. A subroutine called BADIBM induces printout when a non-recoverable error occurs due to improper data or certain machine errors. This subroutine is called when a parameter assumes an unacceptable value.

2.7 Error Correction and Recovery

Two types of error may be encountered in operation. "Non-recoverable" errors cause the computer to print a suitable comment on the output tape and go to the next problem. It skips pseudo instructions until a NEXT instruction is found. If a HALT is encountered first, exit is taken from SAS.

When the engineer wishes to continue calculations in his problem he can re-enter the machine in the usual way except that required tapes must be re-installed on the computer. Any new instructions that the engineer wishes to insert in his program are inserted and those program instructions that have been successfully performed may be omitted if required data is still available for subsequent calculations. Use of the recoverable feature permits the engineer to avoid recalculation.

Errors induced by machine goofs or need to stop operation because of time runout are also treated as non-recoverable errors. On this non-recoverable error the operator transfers to a specific location in the MPC. This transfer provides the usual error printout and success exit. Section 2.4 data provides further explanation of error printouts.

Flow disrupt errors, the second type of error, are those that the engineer may anticipate. These usually are involved with the fact that the matrix being handled is too large for the system. Suitable comments are written on the output tape and the program may follow the sequence of corrective operations required by the engineer. These may include use of more general subprograms or instructions to handle data in partitions.

Table 2-5 lists recoverable errors of the operation subprogram. Any given program has at most one flow disrupt error. When non-recoverable errors occur, the cause is noted on the output tape.

TABLE 2-5

FLOW DISRUPT ERRORS

Subprogram	Cause of Error
MULT	Neither A or B fit in core
CODE	Too many codes in the precoded matrix to fit in core (15,000)
DECO	Same as above (CODE)
COLD	Matrix is not in coded format
ROWS	Matrix is not in coded format
CHOL	A is not positive definite

Master intelligence checks the size of the matrices of all other subprograms to see that matrices required to fit into core actually do fit. If not a flow disrupt error is also signalled.

3. Master Program Control (MPC)

Master Program Control contains master intelligence and the operation subroutines. Master intelligence contains the primary flow control instructions that direct the problem solution. These operations include positioning tapes, assigning storage, transferring library subroutines, handling the interpretation of the pseudo instructions and providing the basic error and logic routines of the program. The flow control instructions, also called logic subroutines (open), which direct these operations, are described in a separate write-up entitled MASTER INTELLIGENCE. These subroutines and the operation subroutines (closed) are listed in Table 3-1.The operation subroutines are also described in separate write-ups. The PREP, VARY, and BACK logic subroutines are used by MAKER in the generation of a program. They are described in the MAKER write-up. They are replaced by other pseudo instructions and are not interpreted by MPC.

TABLE 3 -1

MASTER PROGRAM SUBROUTINES

Logic Subroutines (open)		Operation Subroutines (closed)
PREP	ERRS	MAKER
VARY	SKIP	FINDS
BACK	NEXT	ERROR
	PAWS	
	STOP	TAPES
	HALT	COINS

The separate write-ups for MASTER INTELLIGENCE and the Operation Subroutines follow. These writeups define program purpose, restrictions, method, usage, and coding information.

Master Object Program: MASTER INTELLIGENCE, Flow Control Program

Purpose: To make accessible required data, to select required operation subroutines, and to interpret flow control pseudo instructions.

Restrictions:

(1) Instructions must be included in the SAS pseudo instruction table

(2) Data handling limitations are those imposed by the operation subroutines and operation subprograms used.

Method: Library Tables are read and control is transferred to MAKER to read pseudo instructions and to write the program tape. Each pseudo instruction is read and executed in turn until the program is completed.

There are two types of pseudo instructions--flow control instructions and operation instructions. The flow control instructions include ERRS, SKIP, STOP, NEXT, PAWS and HALT. These are treated directly by MASTER INTELLIGENCE. They cause the following actions:

(1) ERRS: If the previous program resulted in a flow disrupt error, this code directs continuation of calculations. All instructions between the ERRS and succeeding SKIP or STOP will be executed in sequence. If a flow disrupt error has not occurred, all these instructions are disregarded.

(2) SKIP: If in the flow disrupt mode, the number of pseudo instructions denoted in the last field are ignored and those following executed. If not in the flow disrupt mode, this instruction is ignored.

(3) STOP: If in the flow disrupt mode, all succeeding instructions are ignored until a NEXT code is found. If not in the flow disrupt mode, this instruction is ignored.

(4) NEXT: This instruction indicates the start of another case. No calculations are performed.

(5) PAWS: The calculations are stopped. They can be restarted by pressing the start button.

(6) HALT: The CALL EXIT instruction is executed.

MASTER INTELLIGENCE actions under an operation instruction are locating data, calling the operation subroutine, and providing error control. Data is handled in accordance with the subprogram designation as follows: (See coding information Item (6) of the general operation subprogram NAME in Section 5).

(1) Any matrix in core but not permitted in core is erased.

(2) All required matrices in core are retained. They are moved to the upper part of core.

(3) All matrices not required but available and permitted in core are moved to the bottom of core.

(4) Any matrices required in core but not in core are read in from tape behind those required and already in core.

(5) Tapes are positioned for all output matrices and input matrices not in or required in core.

The operation subroutine is called by a CHAIN instruction. (See Section 2.4. for a description of CHAIN). After execution control is returned to MASTER INTELLIGENCE. If necessary, an error printout is made using ERROR.

Usage:

(1) This is an object program.

(2) This program uses MAKER, TAPES, COINS, FINDS, TSNAM, and ERROR as subroutines.

Coding Information:

(1) This program includes SHIFT, IN, and SPACE as subroutines. Their functions are respectively to move data in core, to read data into core, and to provide space in core for the operation subprograms.

(2) This program and its subroutines are maintained in core at all times.

(3) Use of the CHAIN instruction and instructions indicated by comment cards may not be admissible in all compilers.

Operation Subroutine: MAKER, Pseudo program generator

Purpose: To generate and write a set of pseudo instructions on the program tape from a set of input instructions. This program eliminates the instructions PREP, VARY, and BACK by developing additional instructions and checks pseudo instruction logic. It prints out input pseudo instructions and optionally prints out the pseudo program.

Restrictions:

(1) Pseudo instructions must be in the proper format

(2) The maximum number of instructions between a PREP and a BACK must be < 2083.

(3) The last pseudo instruction must be a HALT

(4) The total number of pseudo instructions in the pseudo program must not exceed storage on one tape.

Method: Pseudo instructions are read from the input tape and rewritten on the pseudo program tape until a PREP is encountered. Instructions, after a PREP, are accumulated in core until a BACK is found. Within this range, pseudo instructions are generated for each time through the loop. Address modifications require by VARY are made. After the loop, card reading and writing continues until the HALT is found. Logic checks include checking that each PREP is followed by a BACK before a NEXT or HALT is encountered, that no PREP lies between an ERRS and its companion SKIP or STOP card, that two consecutive PREPS are separated by a BACK that for each instruction there exists a subroutine, and that pseudo card sequence numbers never diminish. If instructions for a given case are in error, that part of the program tape is not written and the next case is processed. On finding errors in the pseudo instruction input, suitable off-line comments are made. If the only error is in the sequence numbers of the pseudo instructions, program progress is not halted.

Usage:

(1) Calling sequence is CALL MAKER

(2) Upon calling, the input tape must be positioned just before the first pseudo instruction.

(3) The error exit is taken only if no satisfactory cases are included in the input.

(4) Pseudo instruction card format is F8.1,2((3X,I5), (2X,A3,I3)), (4X,A4), (3X,I5), (2X,A3,I3), (3X,I5). Data has the following interpretation

(F8.1):	Card sequence number
(3X,I5):	"A" matrix storage assignment

(2X,A3,I3):	A matrix identification
(3X,I5):	"B" matrix storage assignment
(2X,A3,I3):	B matrix identification
(4X,A4):	Pseudo code
(3X,I5):	"C" matrix storage assignment
(2X,A3,I3):	C matrix identification
(3X,I5):	Control information

(5) Print out of the pseudo program will be activated if the control information in the HALT instruction is non-zero.

Coding Information:

(1) This subroutine includes, PR, VA, RE, PT, WR, and BS as sub-functions. Their functions are respectively to prepare a loop, to process a VARY, to read pseudo instructions from the input tape, to check sequence and to print pseudo instructions on the output tape, to write instructions on the program tape and check codes, and to backspace the program tape over an erroneous case. No other subprograms of SAS are required.

(2) This subroutine is used only at the beginning of SAS execution and may be erased thereafter.

(3) The pseudo program tape is left in rewind status upon exit from MAKER.

Operation Subroutine: FINDS, Tape Search

Purpose: To position a tape for reading or writing by an Operation Subprogram.

Restrictions:

(1) Tape group numbers must be in sequence and increase as the tape is moved in the forward direction.

(2) The block number of the last block in each group must be negative and the same as the number of blocks. Block numbers of all other blocks in the group must be positive and in sequence starting from one.

(3) Data must be written on tape in the same form as written by TAPES.

Method: The tape to be searched is backspaced one block and the block read. If the tape group number is less than that desired, the tape is moved forward until it is positioned just before the group required. Completion of the reading of a group is determined by reading a negative block number. If the tape group number is greater than that desired, the tape is backspaced over the groups and positioned in front of the group required. Backspacing is controlled by the block number in the last block of each group. After backspacing over a group, the last block in the preceding group is read to determine how many blocks exist in the next group.

Usage:

(1) Calling sequence is CALL FIND (L,I) L indicates the tape group number being searched. It is a negative five digit number. The upper two digits are the logical tape number. I is the matrix referred to. If I = IA the A matrix tape is being positioned; if I = IB, the B matrix tape, and if I = IC, the C matrix tape.

(2) This subroutine uses TAPES as a subprogram.

Coding Information:

(1) Tape search can be started from any tape position between the beginning of file of the tape and the end of record gap immediately following the last block of the last group.

(2) Tape rewind is used only if positioning is to be before the first group.

(3) If the last block of a group does not contain a negative block number, the non-corrective error exit is taken.

Operation Subroutine: ERROR, Diagnostic Printout

Purpose: To identify the type of error and state of the calculations when an error is encountered.

Restrictions: None

Method: A test of the error indicator (NCE) is made: if positive, a non-corrective error is printed; if negative, a flow-disrupt error is noted. In any case, the pseudo instruction under which the error occurred is printed. In addition, identification information for the A, B, and C matrices is printed if available.

Usage:

(1) Calling sequence is CALL ERROR

(2) Direct transfer to this subroutine can be used at any time to print out the operative pseudo instructions and the matrix identification information. For this purpose instructions are included using ERROR as a subroutine.

Coding Information:

(1) This subroutine is kept in core all during pseudo program execution.

(2) There are no error steps in this subroutine.

Operation Subroutine: TAPES, Tape read-write

Purpose: To handle all block reading and writing on matrix and scratch tapes, tape backspacing, and rewind.

Restrictions:

(1) All blocks on tape must be 127 words long. The first 120 words are data, the last seven words contain data identification.

(2) Tape numbers used must conform with SAS and local tape assignments.

Method: Blocks may be handled in groups or singly. Under group operation, blocks are read into or written from consecutive core locations. Identification information is simultaneously changed in the matrix identification region in core (TRA) as the blocks of data are read or written. In writing, all blocks are written and the last given a negative block number. In reading, as many blocks are read as in the group or the number specified, whichever is least. In individual operation, each block is written or read by separate subroutine entries. Block numbers are increased from the initial value in the TRA block and the final one is set negative.

Usage:

(1) Calling sequence is CALL TAPES (NT, NB, IL, A, J)

NT is the logical tape-group number $1000 \leq NT \leq 14999$. ($NT =$ (Tape No. $\times$ 1000) + group number) Only the thousands digits are used as tape designations. If $NT < 0$, tape is moved backwards; if $NT > 0$, forwards.

NB is the maximum number of blocks to be handled. If $NB < 0$, blocks are to be read. If $NB > 0$, blocks are written. If $NB = 0$ rewind is performed. If $NT < 0$ and $NB > 0$, tape is backspaced NB blocks. $NB \leq 32000$ for a group.

IL is the index value of the first word of identification information for the matrix being handled. If $IL < 0$, blocks are being handled individually, if $IL > 0$ blocks are being handled in groups.

A is the array to be written or read.

J is the index of the first element in the array to be written or read.

(2) This subroutine requires COINS as a subroutine.

(3) Data and identification information must be stored in consecutive locations starting at the initial location. 120 words are taken or placed in core for each data block. The seven words of identification on tape correspond to the first eight matrix identification words in the TRA region.

Coding Information:

(1) After reading a block, the block sequence number of the last block read is left in the TRA region. After writing a block, the block number in the TRA region is increased by one unless the last block of a group has been written. In that case the block number is left unchanged.

(2) There are no non-corrective or flow disrupt steps in this subroutine.

(3) This program can be modified to convert from tape to disk storage any matrix (non-scratch) tape. It must also be modified if logical tape numbers are replaced.

Operation Subroutine: COINS Code Interpreter and Coiner

Purpose: To form or disassemble a packed word to or from integer words defining row and column number.

Restrictions:

(1) Any node number cannot exceed 4095, but can be of plus or minus sign.

(2) The maximum number for nodal components must be less than 9.

(3) All packed words must be positive, all nodal component numbers must be positive.

Method: In coining from four words, sixteen is added to the component number and the sign of the node number changed if it is negative. Packing and unpacking is done by adding and shifting. 12 bits are reserved for node number and five for component number. These 17 bits are treated together when two word packing or unpacking is performed.

Usage:

(1) Calling sequence is CALL COINS (J,I)
J is the first location containing input data (code numbers)
I is the mode index and is interpreted as follows:
-1, coin a one word code from 4 words of input
0, coin a one word code from 2 words of input
1, decompose one word into 2 words; a row and a column code
-2, decompose one word into 4 words; a row node number, a row component number, a column node number, and a column component number.

(2) This is a closed subroutine

(3) Input words and output words must be stored in consecutive locations starting at J(1). A maximum of four words are required.

Coding Information:

(1) This is a FAP subroutine.

(2) This program must be changed if the 35 bit word size is changed.

(3) The program has 96 instructions. If the four word coining and interpretation is separated, there will remain but 46 instructions.

(4) Normal operation will require an average of 32 machine cycles from entry to the linkage until return to the object program.

(5) No storage spaces are required other than those occupied by input, output, and the program.

(6) If a node number is greater than 4095 in absolute value, J(5) is set equal to one before return and the offending code put in J(1). If both node numbers are minus zero, the code is taken as zero.

4. Pseudo Instructions

The function of the pseudo instructions is to provide the user with a simple system through which he can direct calculations. The burden imposed on the analyst by this device is more than repaid by the flexibility of computer operation. The analyst can easily direct any sequence of operations he requires. Thus he can incorporate equilibrium checks and calculation of reactions. Reorganization of the data is at his disposal.

There are two types of pseudo instructions: logic instructions and operation instructions. The logic instructions direct program control. The operation of these instructions when they are acting as logic subroutines is described in Section 3. The operation instructions call for subprograms which operate on matrices. The format for both types of instruction is the same and is shown in Figure 4-1. Data is located in nine fields. The "O" field defines the instruction sequence number. Data in the A,B,D, and E fields is in fixed decimal form. The A, B, and D fields contain the tape storage assignment (tape-group number) for the A, B, and C matrices while field E contains control information for the operation to be performed. Fields 1, 2, and 3 contain the 6 character alphanumeric names of the A, B, and C matrices respectively. The last three characters must be numeric. The C field contain the four letter mnemonic code which specifies the subroutine or subprogram to be used.

Field 0	A	1	B	2	C	D	3	E
F8.1	3X,I5	2X, A3,I3	3X,I5	2X, A3,I3	4X,A4	3X,I5	A3,I3	3X,I5

PSEUDO INSTRUCTION FORMAT

Fig. 4-1

The pseudo instruction shown in Figure 4-2 is a logic instruction. This card is required to cause a transfer of control to the success exit. For some logic instructions data is required in all fields

A list of the logic pseudo instructions and the program interpretation is given in Table 4-1.

Field	0	A	1	B	2	C	D	3	E
	14					HALT			

HALT PSEUDO INSTRUCTION

Fig. 4-2

TABLE 4 - 1

LOGIC PSEUDO INSTRUCTIONS

Instructions	Logic Pseudo Instructions
PREP	Prepare for multiple execution of the following instructions
VARY	Vary matrix or tape numbers in the next instruction by augmenting respective fields by these specified amounts.
BACK	Back up and repeat instructions after PREP
ERRS	Disrupt errors can be corrected as follows
SKIP	Skip the next "n" pseudo instructions (Skip cannot be included between a PREP and BACK instruction
NEXT	This is the first instruction of a new case
STOP	Stop this case and go to NEXT to find the next case
HALT	Halt and transfer to the success exit
PAWS	Pause

The pseudo instruction shown in Fig. 4-3 is an operation instruction. This instruction requires that the A matrix, called MAT001 and stored as group 1 on tape 10, be added to the B matrix, called MBT002 and stored as group 1 on tape 9. The result is the C matrix, called MCT001 and is to be stored as group 1 on tape 12.

It is seen that each matrix is specified by two fields of data. The first field defines the tape and tape group number. If this first word is left blank, the matrix is to be found or left in core. The second field contains letters and numbers which identify the matrix. The letters can be selected by the user. The numbers can be selected by the user except that stiffness, stress and loading matrix numbers are chosen by the program to coincide with the element numbers.

Instructions are performed in the sequence in which they are introduced. Since the instructions do not define the matrix size, they can be used for any problem requiring the same instructional sequence.

A list of the operation pseudo instructions and their interpretation is given in Table 4-2. Further details on the programs evoked may be found in Section 5.

	0	A	1	B	2	Code	C	3	E
2c	3.5	10001	MAT001	9001	MBT002	ADDS	12001	MCT001	
	1	2	3 4	5	6 7	8	9	10 11	12

ADDS PSEUDO INSTRUCTION

Fig. 4-3

TABLE 4 - 2

OPERATION PSEUDO INSTRUCTIONS
(Subprograms)

Instructions	Interpretation
ADDS	Form $[C3] = [A1] + [B2]$
SUBS	Form $[C3] = [A1] - [B2]$
MULT	Form $[C3] = [A1]\ [B2]$
ROOT	Find latent roots of $[A1]$, a symmetric matrix
CHOL	Form $[C3] = [A1]^{-1}\ [B2]$ using Choleski decomposition
CHIN	Form $[B2]$ such that $[B2]\ [B2]^{T} = [A1]$ and $[C3] = [B2]^{-1}$ where $[A1]$ is symmetric and positive definite
ITER	Form $[C3] = [A1]^{-1}\ [B2]$ using accelerated Seidel iteration. $[A1]$ must be positive definite.
WASH	WASH $[B2]$ elements from $[A1]$ eliminating rows and columns to produce $[C3]$
FLIP	Form $[C3] = [A1]^{T}$
SORT	Sort a matrix $[A1]$ by row and column (row listing) as $[C3]$
CODE	Transform $[A1]$ to coded format as $[C3]$
DECO	Transform $[A1]$ to precoded format as $[C3]$
COLS	Put the $[A1]$ matrix in column listing and call it $[C3]$
ROWS	Put the $[A1]$ matrix in row listing and call it $[C3]$
BILD	Construct stiffness, stress, and/or loading matrices as $[A1]$, $[B1]$ and $[C1]$ respectively
READ	Read matrix $[A1]$, $[B2]$, and/or $[C3]$ from cards
INKS	Print matrix $[A1]$, $[B2]$, and/or $[C3]$

Table 4-3 is an example of the pseudo instructions for a pseudo program. The following is an explanation of operations directed by the pseudo instructions.

Instruction 1.0, READ:

This instruction causes the input tape to be read and the information stored as designated on the card. In this case the matrices CAT001, CAT002, and CAT003 are stored on tapes nine, ten and eleven, each as group one.

Instruction 3.0, READ:

This instruction stores matrices CAT004, CAT005, and CAT006 on tapes nine, ten, eleven as group one respectively. Note that though instruction 2.0 is omitted this program is acceptable since instruction numbers never decrease.

Instruction 4.0, ADDS:

This instruction causes matrix CAT001, which is stored on tape 9 as group 1, to be read into core. Then CAT002, which is stored on tape 10 as group 1, is read into core block by block and the elements added to those of the A matrix already in core. The elements are properly matched by use of the row and column codes. The result is called SAT001 and is kept in core.

Instruction 4.1, PREP:

This instruction tells the pseudo program generator MAKER to set up the coding to perform a loop four times.

Instruction 4.2, VARY:

This instruction tells the pseudo program generator MAKER to set up the coding to increment the parameters in the same locations on the next card by the amount indicated on the VARY card minus one. On the first pass of the next card in the loop no incrementation of parameters is made. The parameters are then incremented on each succeeding pass.

Instruction 4.3, ADDS:

This instruction, on the first pass adds SAT001, which is in core, denoted by no tape-group number, to CAT003, which is stored on tape 11 as group 1, and calls the result SAT002.

Instruction 4.4, BACK:

This instruction tells the pseudo program generator MAKER to set up the coding to continue the loop operation started by the PREP instruction until the indexing is complete.

On the second pass through the ADDS instruction, SAT002 is added to CAT004 stored in core and called SAT003. This process continues until the indexing is complete.

Instruction 5.0, INKS:

This instruction causes SAT005, the final result of the indexing, to be printed on the output tape.

Instruction 6.0 HALT:

This instruction "wraps up the program" and returns control to the job system of the computer.

TABLE 4.3

EXAMPLE PSEUDO INSTRUCTIONS

Field	0	A	1	B	2	C	D	3	E
	1.0	09001	CAT001	10001	CAT002	READ	11001	CAT003	
	3.0	09002	CAT004	10002	CAT005	READ	11002	CAT006	
	4.0	09001	CAT001	10001	CAT002	ADDS		SAT001	
	4.1					PREP			4
	4.2			00001	000001	VARY		000001	
	4.3		SAT001	11001	CAT003	ADDS		SAT002	
	4.4					BACK			
	5.0		SAT005			INKS			
	6.0					HALT			

5. Operation Subprograms

The operation subprograms perform operations on the matrices, read input and write output, and then return control to MPC. The following is a list of subprograms, their input and output formats, and the scratch tapes they use if any.

TABLE 5-1

OPERATION SUBPROGRAM FORMATS

Subprogram	Input Format	Output Format	Scratch Tapes Used
ADDS	Coded	Coded	Tape 8 will be used if core cannot contain results.
SUBS	Coded	Coded	Same comments as ADDS
MULT	Coded	Coded	Tape 8 may be used
ROOT	Precoded	Precoded	None
CHOL	Coded	Coded	Tapes 7 and 8 are both used
CHIN	Coded	Coded	None
ITER	Coded	Coded	Tapes 7 and 8 are both used
WASH	Coded	Coded	None
FLIP	Coded	Coded	Tape 8 may be used
SORT	Coded	Coded	Tape 8 may be used
CODE	Precoded	Coded	None
DECO	Coded	Precoded	None
COLS	Coded	Coded	None
ROWS	Coded	Coded	None
BILD	Element Data	Coded	None
READ	Coded or precoded	Coded or precoded	None
INKS	Coded or precoded	Coded or precoded	None

Most of the operation subprograms use the (DAT) region in COMMON storage to store matrices. They also use the other parameters in COMMON storage as switches in directing subprogram flow.

The characteristics shared by all the operation subprograms are grouped together in a separate description entitled NAME on the following pages. This is not a subprogram but a variable title which can apply to any of the operation subprograms. The individual write-ups for the operation subprograms follow the write-up entitled NAME.

Operation Subprograms: NAME, Title of any Individual Subprogram
NAME is not a specific subprogram. It is a variable title referring to all the other subprograms.

Purpose: Description of program functions which are common to nearly all of the subprograms.

Restrictions:

(1) Single-precision real term floating binary arithmetic is used.

(2) Data formats must be coded or precoded and row or column listed. Input matrices must be sorted, row listed, and coded unless stated otherwise.

(3) A, B, and C matrices must be on different tapes unless one is known to be in core and no tape assignment given.

Method: Format, listing, and shape requirements are checked. All these inconsistencies are reported before the non-corrective error exit is taken. If core is too small to handle the data, the flow disrupt error is taken. If the matrices fit in core, all operations are done in core and, if possible, the result left in core and optionally written on tape. The procedure for data handling when core is too small is described where appropriate. Errors in the data discovered in the course of calculations (such as non-positive-definiteness and failure of iteration convergence) are treated as non-corrective.

Usage:

(1) Calling sequence is CALL CHAIN (I, 3) where I is the subprogram identification number and 3 the structural analysis system (SAS) program tape.

(2) Subroutines and subprograms referred to include only those in SAS. Most subprograms use COINS and TAPES as subroutines.

Coding Information:

(1) COMMON contains the following one-dimensional arrays:

ABA: 20 possible logic pseudo codes
ABB: 100 possible operation codes
BUR: The 12 words of the current pseudo instruction
NCE: The 1 word error indicator
TRA: The 30 words for the A, B, and C matrix identification data.
KOD: The 5 word region for forming and decomposing codes
NLD: One word defining the dimension of DAT
NBD: One word defining the number of blocks in DAT
REE: The 120 word storage region for reading tape blocks
WRY: The 120 word storage region for writing tape blocks
DAT: The region for data storage containing NLD locations

COMMON (each subprogram identifies only those arrays addressed by that particular subprogram).

(2) BUR must not be changed by a subprogram. Subprograms are concerned with only the integer data in BUR (2), BUR (5), and BUR (9) and BUR (12). These are the tape numbers of the A, B, and C matrices and the subprogram control information respectively.

(3) NCE is set zero before entry to the subprogram. Upon return, a non-corrective error is indicated by NCE = 1. A flow disrupt error is indicated by NCE = -1. NCE = 0 indicates no error.

(4) TRA is modified by the subprogram to characterize the subprogram output data. This array consists of three groups of ten words. The first eight words in each group are the matrix identification data. These words have the following form and interpretation:

1. Alphabetic: Matrix letter designation
2. Integer: Matrix numeric designation
3. Integer: Number of data blocks in the matrix
4. Integer: Number of rows in the matrix
5. Integer: Number of columns in the matrix
6. Integer: Listing indicator -1, row listed; 0, unsorted; +1 column listed
7. Integer: Coding format 0, coded; 1, precoded
8. Integer: Tape group number
9. Integer: Index locating data in DAT.
   If > 0, matrix is in core at DAT (index value).
   If = 0, matrix is not in core and tape is not positioned for reading.
   If < 0, matrix is on tape, tape is positioned, and matrix can be read into DAT (-index value).
10. Integer: Initial TRA index of identification data for a matrix TRA (10) locates A matrix identification, TRA (20), B, and TRA (30), C.

Only items 3, 4, 5, 6, 7, and 9 can be supplied by the subprogram.

(5) The output tape is 6. The input tape is 5. These are addressed directly for comments. Scratch tapes (7 and 8) may also be directly addressed, need not be written in block format and must be left rewound upon exit from a subprogram if they are used for intermediate storage.

(6) Each subprogram must be given a designation denoting its input and output limitations. This designation is used by Master Intelligence and consists of four numbers. The first number is the chain identification number; the second, the identity of input matrices required in core upon subprogram entry; the third, the identity of input matrices permitted in core; and the fourth, the number of output matrices required. Input matrices are counted from the left on the pseudo instruction; output matrices from right to left. The identity of input matrices is a binary coded number where bits correspond to the A, B, and C positions respectively. If this is six, for example, the A and B matrices are denoted. If two only, B. This designation appears in the coding information of each subprogram.

(7) The non-corrective error designated as a machine error is used to indicate that a parameter has assumed an impossible value during the course of the calculations.

(8) All subroutines are closed.

(9) Identification of matrices read from tape are checked by the operation subroutine. Coded matrices written on tape by the subroutine must fill the last block or be ended by a zero in the last block.

Operation Subprogram: ADDS, Matrix Addition

Purpose: To form C = A + B where A, B, and C are coded matrices

Restrictions:

(1) Tape 8 cannot be used for A, B, or C

(2) None of A, B, or C need fit in core.

Method: If both matrices are in core, they are moved together and sorted as a one-dimensional array using SORTS. If one matrix is on tape it is read into core by block and mixed with the in core matrix until core is full. The rest of the operation is the same as SORT. If neither matrix is in core blocks of the A matrix are read and Mixed into core (using MIXES). Block reading and mixing continues until core or both matrices are exhausted. The raminder of the method is the same as SORT.

Usage:

(1) Calling sequence is CALL CHAIN (8, 3)

(2) This subprogram uses TAPES and SORTS as subroutines. (See SORT for a description of SORTS). COINS is required but not entered. This is because SORTS uses COINS for some applications, but not this one.

(3) This subprogram may use scratch tape 8 for intermediate storage.

Coding Information:

Subprogram designation is (8, 0, 6, 1)

Operation Subprogram: SUBS, Matrix Subtraction

Purpose: To form C = A - B where A, B, and C are coded matrices

Restrictions:

(1) Tape 8 cannot be used for A, B, or C

(2) None of A, B, or C need fit in core

Method: The procedure is the same as for matrix addition (ADDS). On the first pass through the B matrix, signs of the elements are changed, however.

Usage:

(1) Calling sequence is CALL CHAIN (17, 3)

(2) This subprogram uses TAPES and SORTS as subroutines. (See SORT for a description of SORTS). COINS is required but not entered. This is because SORTS uses COINS in some applications, but not this one.

(3) This subprogram may use scratch tape 8 for intermediate storage.

Coding Information:

Subprogram designation is (17, 0, 6, 1)

Operation Subprogram: MULT, Matrix Multiply

Purpose: To form the matrix product C = AB where A, B, and C are coded.

Restrictions:

(1) The B matrix must be column listed.

(2) Either A or B must fit in core

(3) C must be written on tape

(4) The listing of C depends on whether or not A is in core during the multiplication.

Method: If A and B fit in core, A is multiplied by B and the resulting matrix listed by column on the matrix output tape. If a is in core and B does not fit, A is multiplied by B and the partitioned results written on scratch tape. SORTS is then used to sort the complete C matrix by column and write the result on the matrix output tape. If A does not fit in core, the partitioned C matrix is written on the scratch tape and SORTS used to sort the C matrix and list it by row.

Usage:

(1) Calling sequence is CALL CHAIN (9, 3)

(2) This subprogram uses COINS, TAPES, and SORTS as subroutines

(3) This subprogram may use scratch tape 8 for intermediate storage

Coding Information:

The subprogram designation is (9, 0, 6, 1)

Operation Subprogram: ROOT, Symmetric Matrix Roots

Purpose: To obtain the latent roots and vectors of A, a square, symmetric matrix

Restrictions:

(1) The A matrix must be in precoded format, square, symmetric, and prestored in the first part of the data region.

(2) Matrix order must be greater than 1 and less than 96.

Method: Givens' method (See Modern Computing Methods, Philosophical Library, 1961, p. 30, 31) is used. Matrix shape, size, format and symmetry are checked. Symmetry is required to four significant figures. If the matrix is too large, the flow disrupt error exit is taken. If the roots can be obtained, the matrix is expanded into a two-dimensional array and ANF402, a SHARE FORTRAN subroutine is used to get both the roots and vectors. Vectors are given the B matrix designation; and roots, the C matrix designation. Vectors are of unit length. Row codes of the B matrix are taken to be those of the A matrix. Column codes of the B and C matrices are numbered in sequence. Matrices are rewritten in precoded format and written on tape if required.

Usage:

(1) Calling sequence is CALL CHAIN (15, 3)

(2) This subprogram uses ANF402 (SHARE FORTRAN subroutine) as a subroutine. The TAPES subroutine of SAS is also required.

Coding Information:

(1) Machine time is proportional to matrix order squared. An upper bound on time is $T = (1/20)N^2$ where T is in seconds and N the matrix order.

(2) This routine may fail to locate all vectors for multiple roots. In any case, these vectors are generally not mutually orthogonal.

(3) Underflow may occur with predominately zero or near zero matrices.

(4) A small increase in matrix order handled (one or two orders) can be achieved by rearrangement of erasable storage in conformance with final system storage assignment. If only latent roots are desired, matrix order can be increased to 140 x 146 by a reassembly.

(5) The subprogram designation is (15, 4, 4, 1).

Operation Subprogram: CHOL, Simultaneous Equation Solution Using Choleski Triangular Decomposition

Purpose: To form $C = A^{-1}B$ where A is a square positive definite matrix and B a set of vectors.

Restrictions:

(1) A must be on tape. A must be square and positive definite and is assumed symmetric.

(2) B must be on tape and be listed by column.

(3) C must be written on other than the B tape and will be coded and row listed.

(4) The diagonal and upper triangular part of A must fit in core in cariable band form along with two full columns of the B matrix.

Method: Coded elements of A are read and a table of distinct codes compiled. The coded B matrix is read and written on scratch tape followed by the code table. The A matrix is then read into core and put in variable bandwidth form, a list of the locations of the last element in each row being formed. A is then decomposed such that $U U^T = A$. If square roots of negative numbers are required the matrix is not positive definite and the non-corrective error exit is taken. The equations $L_y = B$ are solved, y replacing B in core. Then, $L^T U = y$ is solved, U replacing y. Each column of B is treated separately and columns of C written on the scratch tape. Finally, the columns of C are read from the scratch tape, coded, and written on the output matrix tape. Columns of C are numbered in sequence.

Usage:

(1) Calling sequence is CALL CHAIN (13, 3).

(2) This subprogram uses COINS and TAPES as subroutines.

(3) This subprogram uses both scratch tapes for intermediate data storage.

(4) The C matrix will be listed by column and will contain elements for every row of the A matrix.

(5) If an extra row is included in the B matrix, a comment is printed, this element is ignored, and calculations are continued.

Coding Information:

(1) Maximum matrix size is 6666 x 6666 with no practical limit on the number of columns in the B matrix. Actual size limits

depend on the average bandwidth by $N = 20000/(3 / b)$ where b is the average number of elements in the band not including the diagonal. The maximum matrix size is attained, for a given problem, by numbering the nodes so that the band is kept to a minimum and increased as slowly as possible. The program is intended to handle the solution of simultaneous equations for structures with stiffness matrices of order 2 x 2 to 500 x 500.

(2) The Choleski algorithm is a subroutine contained within this subprogram.

(3) The non-corrective exit is taken if A is not positive definite or no tape assignment is given for C as well as for format, shape, and listing errors.

(4) The subprogram designation is (13, 0, 0, 1)

Operation Subprogram: CHIN, Form the inverse of the Choleski decomposition matrix

Purpose: To form B = U where U U^T = A, and C = U^{-1}

Restrictions:

(1) A must be on tape. A must be square, positive definite, and is assumed symmetric.

(2) B and C will be coded and listed by row.

(3) The diagonal and upper triangular part of A must fit in core with two vectors of a length equal to the order of the A matrix.

Method: Coded elements of A are read and a table of distinct codes compiled. The "A" tape is backspaced and reread and the matrix put in variable bandwidth form, a list of the last element in each row being formed. A is then decomposed such that U U^T = A. If square roots of negative numbers are required, the matrix is not positive definite and the non-corrective error exit is taken. U^{-1} is formed in core, defining C, and the result coded and may be left in core and optionally written on tape. It will be left in core if it can be contained there.

Usage:

(1) Calling sequence is CALL CHAIN (12, 3).

(2) This subprogram uses COINS and TAPES as subroutines.

(3) Code numbers in C will correspond with those in A.

Coding Information:

(1) Maximum matrix size is 6666 x 6666; actual limit depends on the bandwidth which can be effected by the choice of node numbers in the problem.

(2) The decomposition and inversion is a separable subroutine of this subprogram.

(3) The non-corrective exit is taken if A is not positive-definite as well as for format, shape and listing errors.

(4) Subprogram designation is (12, 0, 0, 1).

Operation Subprogram: ITER, Simultaneous Equation Solution using Accelerated Seidel Iteration

Purpose: To form $C = A^{-1}B$ where A is a square positive definite matrix and B a set of vectors

Restrictions:

(1) A must be square and have non-zero diagonal elements. The iteration will not converge if A is indefinite.

(2) B must be listed by column.

(3) Two columns of C must fit in core simultaneously.

(4) C will be written in coded format, listed by column, but not sorted within the column.

(5) A, B, and C must be written on tape and must be on different tapes.

Method: Core storage is reassigned and tapes repositioned if necessary. As much of A as will fit is read into core, recoded, and diagonals check to be non-zero. The rest of A, if any, is similarly treated by blocks and put on the two scratch tapes. The first block of B is read in. Rows corresponding to the first B block are relaxed and then additional B blocks are read and rows relaxed until the first column of the C matrix has been changed. When either the required accuracy or maximum number of iterations has been obtained, the columns are written in blocks on the output tape with column coding matching the B column designations. A null vector is taken as the initial guess and standard Seidel iteration is used for the first cycle. Successive iterations are performed using an adjusted overrelaxation factor. For the first column of C this is defined as 1.50. When successive estimates of the latent root differ by less than 1/1000, and at least ten percent of the iterations have been performed, a new estimate for the overrelaxation factor is formed using Young's formula ("Iterative Methods for Solving Partial Difference Equations of the Elliptic Type," Trans. Amer. Math. Soc. Vol. 76, p 92).

Usage:

(1) The calling sequence is CALL CHAIN (11, 3).

(2) This subprogram uses COINS and TAPES as subroutines.

(3) Printout of iteration progress may be obtained by using a negative E field input in the pseudo instruction.

(4) If a number is included in the E field, its upper digit defines the number of significant figures of accuracy required and the lower three, the maximum number of cycles of iteration. If the upper digit is zero, 3 significant figures are required. If the lower digits are zero, n/3 iterations are tolerated where N is the matrix order.

(5) Both scratch tapes are used by the program even though matrices may fit in core.

Coding Information:

(1) Maximum matrix size is 10,000 x 10,000. Maximum size for all in core operation is $2N - \frac{1}{k^2} + \frac{40,000}{k} - \frac{1}{k}$ where k is the percent of non zero elements in the matrix.

(2) Recode is written as a subroutine in this subfunction. It recodes the matrix, forming a code table, and checks for zero diagonals. Recoding consists of negative numbers for diagonals and new rows. Numbers are the indices of the C element required in the matrix multiplication.

(3) Column codes of the C matrix are chosen to correspond with those of B.

(4) If convergence is not obtained in the permitted number of cycles, the error exit is taken. This exit is also taken if a diagonal is zero or no tape assignment is given for output.

(5) The subprogram designation is (11, 0, 0, 1).

Operation Subprogram: WASH, Special matrix multiplication

Purpose: To pre- and post-multiply a square matrix by a diagonal matrix where only the non-unit elements are supplied.

Restrictions:

(1) The A matrix must fit in core.

(2) The B matrix need not be sorted.

Method: All A matrix codes are decomposed and checked to see if only diagonal elements are included. Compound codes are replaced by row codes. Blocks of the B matrix are treated, one at a time. Treatment consists of decoding. When neither row or column codes correspond the element is left unchanged. For each code component of a B compound code that matches an A code, the Corresponding elements are multiplied. Zero elements are dropped and the matrix clinched up accordingly. The C matrix is left in core or written on tape as required. Both the A and B matrices are destroyed during calculation.

Usage:

(1) Calling sequence is CALL CHAIN (10, 3).

(2) TAPES and COINS subroutines are required by this subroutine.

Coding Information:

(1) If the B matrix will fit in core, the C matrix will always fit in core.

(2) In addition to format errors, the non-corrective stop is taken when elements of the A matrix are not on the diagonal or if a zero code is introduced before the last element.

(3) A flow disrupt error is taken when A will not fit in core.

(3) The subprogram designation is (10, 4, 6, 1).

Operation Subprogram: FLIP, Matrix Transpose

Purpose: To form C, the transpose of the A matrix

Restrictions:

(1) Neither A nor C need fit in core

(2) A and C cannot be on the same tape

Method: Row and column code designations are exchanged and listing indicator changed. Thus if A is originally row listed, $C = A^T$ is row listed.

Usage:

(1) Calling sequence is CALL CHAIN (5, 3).

(2) This subprogram uses TAPES and COINS as subroutines.

Coding Information:

Subprogram designation is (5, 0, 4, 1).

Operation Subprogram: SORT, Matrix Sort and Union

Purpose: To order the A matrix, a coded matrix, by ascending codes adding elements with the same code.

Restrictions:

(1) The matrix must be in coded format

(2) Neither the input nor the output matrix can be on tape 8.

Method: If the matrix is in core or fit in core, it is sorted and written on tape, if required. If the matrix does not fit into core, core is filled and sorted and the resulting partition written on the scratch tape by block. The remaining blocks are sorted and mixed so that after the entire matrix is passed over the lowest element numbers remain in core. Each sorted block is also written on the scratch tape. The lowest "slab" in core is written from core onto the output matrix tape. The scratch tape is then read a multiple number of times, the slab of the next lowest codes and corresponding elements being extracted in each pass and written on the output tape.

Usage:

(1) Calling sequence is CALL CHAIN (14, 3).

(2) This subprogram uses TAPES as a subroutine. COINS is required but not entered in the SORT option.

(3) This subprogram may use 8 as a scratch tape for intermediate storage.

Coding Information:

(1) This subprogram includes SORTS as a subroutine and MIXES and ORDER as subfunctions of SORTS. ORDER rearranges a one-dimensional array by increasing code. MIXES collates two one-dimensional arrays summing elements in the intersection. SORTS directs flow and handles recoding when required (as in FLIP, MULT, COLS, and ROWS). Calling sequences for these subroutines are as follows:

CALL SORTS (M, N, NI)

M is the sorting indicator
-1, order codes (with elements)
0, interchange row and column numbers in codes before ordering
1, interchange row and column numbers in codes before and after ordering

N is the number of matrices involved (one or two). If $N<0$, -N is the number of blocks on the scratch tape and MULT is the operation.

NI is the union indicator

 1, add matrices A and B
-1, subtract matrices A and B

CALL MIXES (KUT, KUP)

KUT is the highest code for which sorting is complete.

KUP is the index of the last code in the DAT region. New codes may be stored two locations beyond this location upon entry to MIXES.

CALL ORDER (A,L)

A is the array to be ordered

L is the array index of the last code in the array

The subprogram designation is (I4, 0, 6, 1)

Operation Subprogram: CODE, Matrix coding and contraction

Purpose: To put a precoded matrix in coded format or rename a coded matrix

Restrictions:

(1) Input and output matrix tapes must be different tapes

(2) All codes from the precoded matrix must fit in core

(3) Output will be in the same sort as input

Method: If already in coded format the matrix is read from tape, renamed, and written on the output matrix tape, one block at a time. If not, all codes are read into core. Elements are read into core one block at a time. Codes are generated for the non-zero elements from the row and column designations. The output is written on the output matrix tape by block and followed by the identification block.

Usage:

(1) Calling sequence is CALL CHAIN (6, 3)

(2) This subprogram requires COINS and TAPES as subroutines.

Coding Information:

(1) It will be desirable to include the capability of all-core operation in this subprogram to increase execution speed if extensive use is made of this subprogram.

(2) The subprogram designation is (6, 0, 0, 1)

Operation Subprogram: DECO, Matrix decoding and expension

Purpose: To put a matrix in precoded format or rename a precoded matrix

Restrictions:

(1) Input and output matrix tapes must be different tapes,

(2) All codes of the precoded matrix must fit in core.

(3) Output will be in the same listing as input.

Method: If already precoded, the matrix is read from tape, renamed, and written on the output matrix tape. Alternately, the coded matrix is read from tape, a block at a time, a table of unique row and column codes is formed and written on the output matrix tape. The matrix is then reread and elements of the precoded matrix stored in blocks and written on tape. Zero elements are supplied as required.

Usage:

(1) Calling sequence is CALL CHAIN (7, 3).

(2) This subprogram uses TAPES and COINS as subroutines.

Coding Information:

(1) It will be desirable to include the capability of all-core operation in this subprogram if extensive use is made of this subprogram.

(2) The subprogram designation is (7, 0, 0, 1).

Operation Subprogram: COLS, Matrix Column Lister

Purpose: To resort a matrix, if necessary, so it is listed by column.

Restrictions:

(1) Input and output matrix tapes must differ.,

(2) Matrix must be in coded format

Method: If the matrix is already column listed a comment is printed and the matrix written on tape if required. Otherwise, the row and column designations are interchanged in the codes and the matrix sorted. After sorting the codes are returned to their original status.

Usage:

(1) The calling sequence is CALL CHAIN (3, 3).

(2) This program requires TAPES and SORTS as subroutines.

Coding Information;

(1) A flow disrupt error occurs if the matrix is in precoded format.

(2) Most of this routine is used for both ROWS and COLS.

(3) The subprogram designation is (3, 0, 4, 1).

Operation Subprogram: ROWS, Matrix Row Lister

Purpose: To resort a matrix, if necessary, so it is listed by row.

Restrictions:

(1) Input and output matrix tapes must differ.

(2) Matrix must be in coded format.

Method: If the matrix is already row listed a comment is printed and the matrix written on tape if required. Otherwise, the row and column designations are interchanged in the codes and the matrix sorted. After sorting the codes are returned to their original status.

Usage:

(1) The calling sequence is CALL ROW (4, 3)

(2) This program requires TAPES and SORTS as subroutines

Coding Information:

(1) A flow disrupt error occurs if the matrix is in precoded format.

(2) Most of this routine is used for both ROWS and COLS.

(3) The subprogram designation is (4, 0, 4, 1).

Operation Subprogram: READ, Read a matrix from cards

Purpose: To read a matrix from the input tape and store it in core or on tape

Restrictions:

(1) The identification card must precede the matrix.

(2) The matrices must be in the sequence in which they are called by the pseudo instruction.

(3) Matrix date may be in coded or precoded format.

(4) Matrix data must match identification information.

Method: The identification card is read. The matrix number is checked with that required and the matrix data read using the identification information. The matrix is read and stored in core or on tape one block at a time.

Usage:

(1) Calling sequence is CALL CHAIN (1, 3).

(2) TAPES and COINS are used as subroutines.

(3) The identification card must be in format (A3, I3), 5I6.
It contains the following data:
(A3, I3): The alphanumeric identification of the matrix. The numeric designation must be greater than zero.
(I6): The number of input cards in the matrix
(I6): The number of rows in the matrix
(I6): The number of columns in the matrix
(I6): Listing: -1, listed by row; +1, listed by column
(I6): Format: 0, coded; 1, precoded

(4) Coded matrices must be in format (3(2(I5,I1), F12.0)) Data on each card is interpreted as follows:
(I5, I1): The row identification number
(I5, I1): The column identification number
(F12.0): The matrix element

(5) Precoded matrices must have row and column identifications in format (12(I5, I1)) and elements in format (6F12.0).
Codes must precede matrix elements.

Coding Information:

(1) This program contains the subroutine CR which is used to store the matrix in core.

(2) The non-corrective error exit is taken if identification does not match that of the pseudo card or if a zero code is used in a precoded matrix. Format errors will also occur if cards are improperly punched.

(3) The subprogram designation is (1, 0, 0, 3).

Operation Subprogram: INKS, Matrix Printer

Purpose: To print out the A, B, and/or C matrices as required, in either coded or precoded format

Restrictions: (1) Precoded matrices must fit in core.

(2) Header cards are printed out with the "A" matrix only.

Method: All in-core matrices to be printed are printed first. Then all on tape matrices required are printed. For precoded matrices the entire matrix is read into core a block at a time and then printed. Pre-coded matrices are partitioned vertically. Each printed page contains the matrix name, sequential page number, and appropriate row and column headings. The page following the last page with matrix print out contains a statement giving the matrix name and the total number of pages of printout for that matrix.

Usage:

(1) Calling sequence is CALL INKS

(2) This subroutine requires TAPES and COINS as subroutines.

(3) The "Header Card" format is 12A6. Information in card columns 1 to 72 inclusive is printed.

(4) The number of "Header Cards" to be read is entered in the E field of the pseudo instruction card with format I5. The maximum value of this integer is 32767.

Coding Information:

(1) This subroutine double spaces the matrix print out. Comment cards indicate the location of changes necessary to single space the matrix print out.

(2) Data in the DAT region, beginning at the bottom, is destroyed by a precoded matrix read from tape. The precoded matrix will occupy the first locations.

(3) Format statements are not compatible with the Philco 2000.

(4) If the ID of an on-tape matrix to be printed does not agree with the ID of the matrix on the pseudo instruction card, an error print out is made and flow disrupt exit taken after all matrices are handled. If the precoded matrix will not fit in core an error comment is printed out.

6. BILD Subroutines

The purpose of the BILD Subprogram is to control flow between the various matrix generator subroutines. The flow is as follows. On receipt of a BILD command, MPC selects the BILD subprogram from the library tape, reads it into core and transfers control to it. BILD then directs the reading of material tables and input data for each structural element, checking of the data, and selecting the appropriate generation subroutine.

The BILD usage writeup appears on the pages following. An explanation of input formats defined is given in section 7. Section 7.2 describes the interpretation of the material table data required by BILD.

Operation Subprogram: BILD, Generator of Structure Matrices

Purpose: To read material tables and element data and identify element subroutine output and write on matrix tapes.

Restrictions:

(1) The input tape must contain the material tables and element input data and be positioned to read the material tables

(2) All output matrices must be written on tapes.

(3) Only FACET elements can be treated.

Method: The material tables are read from the input tape. Element data for an element is read in, data checked, data optionally printed out, and transfer made to the required generation subroutine. This procedure is repeated for each element. Matrix numeric names are assigned to correspond with the element number. Tape assignments are consecutive starting from the initial assignment of the pseudo instruction and in the order of A, B, C if common tapes are used.

Usage:

(1) Calling sequence is CALL CHAIN (16, 3).

(2) This subprogram uses COINS and TAPES as subroutines. In addition it requires a subroutine for each of the structural types identified by the element data. The only such subprogram currently available is FACET.

(3) The materials table must be ended with a blank card. Data for a given material must be grouped together and ordered by increasing temperature. Material table data is generally in E8.4 format. Two cards are required for each temperature level. Information on these cards is given the following interpretation:

(2X,A6): The material identification
(E8.4): Temperature related to the given properties
(E8.4): Coefficient of thermal expansion, inches/inch degree
6(E8.4): D_{11}, D_{21}, D_{61}, D_{62}, D_{66}, material coefficients, lbs/in^2
9(E8.4): D_{31}, D_{32}, D_{33}, D_{44}, D_{54}, D_{55}, D_{63}, material coefficients lbs/in^2, (second card)

Any of the data may be zero except the material identification.

(4) Element input data must include at least two cards of data. Each card is numbered in the first column in Format I1 with a number from one to six. The first card of data for an element must be numbered one and the last, six. Other cards for the element need not be in sort. Omitted data is assumed to be zero. Format of the first card is (I1, 2X, I3, 2X, I4, 3I2)

A6, 4F6,0) Format of all other cards is (I1, F5.0, 11F6.0). Data is given the following interpretation:

Card	Format	Data
1-6	(I1)	Card number
1	(2X, I3)	Element number
1	(2X, I4)	Structural type
1	(I2)	Coordinate system indicator
1	(I2)	Temperature indicator
1	(I2)	Prestress indicator
1	(A6)	Material identification
1	(F6.0)	Temperature level
1	(F6.0)	Pressure level
1	(E6.0)	Weight level
1	(5F6.0)	Geometric data
2	(F5.0,11F6.0)	Element temperatures and prestresses
3	(F5.0,11F6.0)	Element local reference plane location data
4-6	(F5.0,11F6.0)	Element node numbers and nodal coordinates

(5) The absolute value of the psudo instruction E field input defines the number of elements to be handled. If the number is negative, the material tables and element input are written on the output tape. Matrices will be generated only if a tape assignment is given for them in the pseudo instruction.

(6) An error exit will occur if cards are in improper format or sort.

Coding Information:

(1) If new generator subroutines are to be added, additional branches must be provided in the form indicated in the comment cards.

(2) Subprogram designation is (16, 0, 0, 3).

6.1 FACET Subroutine

The "triangular facet" BILD subroutine is the only generation subroutine included presently. This subroutine is called FACET. It generates stiffness, stress, and loading matrix coefficients for a thin flat triangular element. It performs coordinate transformations required. The chief purpose of this subprogram is to provide the capability to analyze plates and shells. It can be used to approximate any structural geometry, however.

The following information concerns the particular interpretation of element input data by the FACET subroutine. General requirements and definitions of input data are given in Section 7.

1. The "structural type" number is 3001
2. "Pressure level": Normal pressure in the local plus z direction in pounds per square inch.
3. "Weight level": <0 weight in pounds per cubic inch
   >0 weight of facet in pounds
4. "Basic geometric data": Thickness of facet. This occupies Field 8 in Table 7-2 and 7-3.
5. Surface temperatures: The temperatures of the upper and lower surfaces in that order. These two quantities occupy Fields 1 and 2 respectively in Tables 7-2 and 7-3.
6. The local reference system is a rectangular cartesian coordinate system.

Writeup of the FACET subroutine is given on the following pages. Section 7.3 contains a general writeup of the interpretation of element input data. The theoretical basis for the equations used in FACET are included in the SAS technical Report.

BILD Subroutine: FACET

Purpose: To generate and tape the stiffness, stress, and/or loading matrices in coded format.

Restrictions:

(1) Adequate element data must be introduced to define a constant thickness flat triangular shell element.

(2) Coordinate systems must be chosen to be consistent with the rest of the elements.

Method: The FACET package consists of 17 subfunction routines called, as required, by the FACET subroutine. These first generate the facet natural coordinates. In this coordinate system the loading matrix coefficients and the stiffness matrix coefficients are generated. The coefficients of the matrix to transform to the local coordinate system are then developed and the loading and stiffness matrices transformed to this system. The transformation coefficients relating the final and local system are generated. The loading matrix is then transformed to this system, coded, and written as the C3 matrix. The stiffness matrix is postmultiplied by the transformation matrix to put the displacements in the final coordinate system. If required, the transformation matrix coefficients which convert the stiffness into the stress matrix are then calculated and the transformation performed. The stress matrix is written on tape as the B2 matrix. The stiffness matrix is transformed to the final coordinate system and written on tape as the A1 matrix.

All matrices are identified by the alphabetic designation given on the BILD pseudo instruction but are numbered to correspond with the related element number. Matrices are stored in consecutive groups on the tapes designated on the BILD pseudo instruction. Equations used in the program are developed in the Structural Analysis System Technical Report.

Usage:

(1) Calling sequence is CALL FACET
(2) This subroutine uses TAPES and COINS as subfunctions
(3) Element data required is contained on cards one and six. Input on cards two and three may be used but any data on cards four and five will be disregarded.
(4) A nonrecoverable error will occur if the element data does not define a facet.

Coding Information:

(1) The purpose of each of the subfunction programs is as follows:

Subfunction	Purpose
COR	Finds natural coordinates and moments of inertia
TRN	Forms transformation matrix from natural to local and local to final coordinate systems.
LOD	Forms loading matrix coefficients in local coordinates system
STF	Forms stiffness matrix in natural system
STR	Forms matrix to transform stiffness to stress
SFT	Transform stiffness matrix to final coordinates
PATH	Integrates edge loads to define stress
INVER	Inverts a 3 x 3 matrix
ADM	Relocates and adds 3 x 3's to form transformation matrices
SAD	Forms a component of the stiffness matrix
SCA	Multiplies a 3 x 3 matrix by a scalar
ADD	Adds 3 x 3 matrices
MUL	Multiplies 3 x 3 matrices
TABLE	Interpolates in the material tables for element properties
MOD	Modifies the transformation matrix for local coordinates
COES	Codes and writes matrices on tape
ADS	Stores 3 x 3 matrices in a 15 x 15 array

(2) This subprogram uses the DAT region for its erasable storage.

(3) See the information on the following four pages.

TABLE 6-1

TABLE OF CODE - COMPONENT NUMBERS

Stiffness Matrix Row/Column Codes;
Stress Matrix Column[*] Codes; Loading Matrix Row Codes

Row/Column	Code	Component	Symbol	Description
1	First Node Number	1	x1	Displacement x-direction; first node
2	"	2	x1	Rotation about x-axis; "
3	"	3	y1	Displacement y-direction; "
4	"	4	y1	Rotation about y-axis; "
5	"	5	w_1	Displacement z-direction; "
6[**]	"	6	z1	Rotation about z-axis; "
7	Second Node Number	1	x2	Displacement x-direction; second node
8	"	2	x2	Rotation about x-axis; "
9	"	3	y2	Displacement of y-direction; "
10	"	4	y2	Rotation about y-axis; "
11	"	5	w_2	Displacement z-direction "
12[**]	"	6	z2	Rotation about z-axis; "
13	Third Node number	1	x3	Displacement x-direction third node
14	"	2	x3	Rotation about x-axis "
15	"	3	y3	Displacement y-direction; "
16	"	4	y3	Rotation about y-axis "
17	"	5	w_3	Displacement z-direction "
18[**]	"	6	x3	Rotation about z-axis "

[*] Stress matrix has an additional column (thermal stress) with a code of zero and a component number 9.

[**] These rows are deleted if the node is considered in local coordinates. In this case all following row numbers are reduced by one. Dimensions for the stiffness matrix may vary from 15 x 15 to 18 x 18, for the stress matrix from 8 x 16 to 8 x 19, and for the loading matrix from 15 x 5 to 18 x 5.

TABLE OF CODE - COMPONENT NUMBERS

Stress Matrix Row Codes

Row	Code	Component	Symbol	Description
1	Element Number	1	σ_{11}	Normal force/length in x-direction
2	"	2	σ_{22}	" y-direction
3	"	3	σ_{12}	Shear force/length in xy-plane
4	"	4	M_{11}	Bending moment/length about x-axis
5	"	5	M_{22}	" y-axis
6	"	6	M_{12}	Twisting moment about xy-plane
7	"	7	V_1	Shear force/length in yz-plane
8	"	8	V_2	" xz-plane

TABLE OF CODE - COMPONENT NUMBERS

Loading Matrix Column Codes

Column	Code	Component	Description
1	0	1	Gravity loading applied in x direction
2	0	2	" y direction
3	0	3	" z direction
4	0	4	Pressure loading in local z direction
5	0	5	Thermal loading

Coding Information:

(1) The purpose of each of the subfunction programs is as follows:

Subfunction	Purpose
NAR	Finds natural coordinates and moments of inertia Forms transformation matrix from natural to local and local to final coordinate systems.
LOD	Forms loading matrix coefficients in local coordinates system
STF	Forms stiffness matrix in natural system
STR	Forms matrix to transform stiffness to stress
SFT	Transform stiffness matrix to final coordinates
PATH	Integrates edge loads to define stress
INVER	Inverts a 3 x 3 matrix
ADM	Relocates and adds 3 x 3's to form transformation matrices
SAD	Forms a component of the stiffness matrix
SCA	Multiplies a 3 x 3 matrix by a scalar
ADD	Adds 3 x 3 matrices'
MUL	Multiplies 3 x 3 matrices
TABLE	Interpolates in the material tables for element properties
MOD	Modifies the transformation matrix for local coordinates
CODES	Codes and writes matrices on tape
ADS	Stores ~~3 x 3 matrices in a 15 x 15 array~~

(2) This subprogram uses the DAT region for its erasable storage.

7. Input Data

This section describes the four principal types of input data: pseudo instructions, material tables, element data, and matrix data. Heading cards form a fifth type of input but are described in Section 2.4 and in Section 5 in the INKS writeup. This section is introduced with a brief explanation of pertinent FORTRAN formats.

7.1 Explanation of FORTRAN Formats

I (Integer) Format: The form IW specifies an integer field w columns wide, where "w" includes the digits, any blanks, and a sign. The integer value itself must be less than 32768. The field is always right justified, e.g., 4987 written in format I6 would appear as ΔΔ4987.

F (External Fixed Point) Format: The form Fw.d specifies a decimal field w columns wide, where "w" includes a sign, the digits, a decimal point, and any blank and "d" specifies the number of digits after the assumed decimal point. When F format is used for input the decimal point is optional. However, if it is inserted, it overrides the "d" specification. Example: If the input is F8.4 the number -2.376 could be punched as -2.376. In this case the three place decimal overrides the d = 4 specifications of the F field. 382 could be written as Δ3820000 for format F8.4 with the decimal omitted.

In using an F format the size of the output numbers must be known so that the "w" specification will encompass at least the integer portion of the number. Otherwise leading digits will be dropped.

<u>E (Floating Point) Format</u>: The form Ew.d specifies a floating point field w columns wide, where "w" includes a sign, the digits, a decimal point, the exponent sign, the exponent, and any blanks. The "d" specifies the number of digits after the decimal point, not counting the exponent and its sign. For input a decimal point is optional in the mantissa. If it is used it overrides the "d" specification. Example: E12.4, 498.72 could be punched on the card as 49672E-02 or as 496.72E+00. The exponent cannot exceed ±38 in magnitude. There are several ways to write E formats. A FORTRAN manual* should be consulted to take advantage of several shortcuts available.

<u>A (Alphanumeric) Format</u>: The form Aw specifies an alphanumeric field w columns wide. The characters can be any of the characters acceptable to the computer including blanks. This is the format used to read in header cards, matrix identifications and instruction names.

<u>X (Space) Format</u>: The form wX specifies a field of w spaces (blanks) wide.

7.1 Pseudo Instructions

Pseudo instruction format, interpretation and examples are described in Section 4.

*Daniel D. McCracken, "A Guide to FORTRAN Programming," John Wiley & Sons, Inc., New York.
"Reference Manual 709/7090 FORTRAN Programming System," C28-6054-2 IBM Corporation

7.2 Material Tables

The material tables define the mechanical properties of all materials of interest. When new materials are to be considered they can be added to the table. The only limitation on the number of materials that may be included is that the table can only involve a total of 398 cards.

Most material table data is entered as floating decimal input in format E8.4. Two cards must be entered for each material. Data can be entered in the table for as many temperatures as desired. For a given material, the cards must be sorted so that temperatures increases from one pair to the next. If data is required at a temperature not given in the tables, then BILD will extrapolate or interpolate the material properties using linear interpolation and data for the two closest temperatures. If material properties are given for only one temperature and data is wanted at another temperature BILD will use the material properties for the temperature appearing in the table. A single zeros [redacted] card is used to signify the end of the table. This blank card is not included in the above limitation of 398 cards in the material table.

Table 7.2 defines the format and interpretation of the two cards required for each material entry.

TABLE 7-1

MATERIAL TABLES INPUT DATA

Card	Field	Format	Information
1	1	(2X, A6)	The material identification: Each material must have a unique identification number. It is recommended that standard SAE and Aluminum Association numbers be used insofar as possible.
1	2	(E8.4)	Rankin temperature for the given material properties.
1	3	(F8.4)	Coefficient of thermal expansion: inches/inch-degrees Rankin
1	4-9	(6E8.4)	Material stiffness coefficients; D_{11}, D_{21}, D_{22}, D_{61}, D_{62}, D_{66}, lba/sq.in.
2	1-7	(7E8.4)	Material stiffness coefficients. D_{31}, D_{32}, D_{33}, D_{44}, D_{54}, D_{55}, D_{63}, lbs/sq.in.

The material stiffness coefficients enter into the stress-strain equations in accordance with the equations.

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xz} \\ \sigma_{yz} \\ \sigma_{xy} \end{Bmatrix} = \begin{bmatrix} D_{11} & & & & & \\ D_{21} & D_{22} & & & \text{SYMMETRIC} & \\ D_{31} & D_{32} & D_{33} & & & \\ 0 & 0 & 0 & D_{44} & & \\ 0 & 0 & 0 & D_{54} & D_{55} & \\ D_{61} & D_{62} & D_{63} & 0 & 0 & D_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{xz} \\ \epsilon_{yz} \\ \epsilon_{xy} \end{Bmatrix}$$

These define a material with at least one plane of elastic symmetry (the x-y plane). If values of D_{61}, D_{62}, D_{63}, and D_{54} are assigned zero an orthotropic material (three mutually orthogonal planes of symmetry) can be defined. If, in addition $D_{44} = D_{55} = D_{66}$, $D_{11} = D_{22} = D_{33}$, $D_{21} = D_{31} = D_{32}$ and $D_{44} = D_{11}/2(1 + D_{21})$ an isotropic material is described.

7.3 Element Data

The purpose of the element data is to identify the element and define its state and geometry. Element data consists of at least two cards and at most six cards of input per element. The two required cards are the first and sixth. The first card contains control and identification data. The sixth card defines the coordinates of the first three nodes of the element. The other cards define the local geometry, relation of overall geometry to the overall rectangular cartesian coordinate system, surface temperatures, pre-stresses, and nodal coordinates. The number of words in the input is fixed to handle the most general element, a prism. The last eight columns of input may be used by the operator for identification of his data. They are not read or used during the computations. Cards one and six must be first and last respectively, but cards two, three, four and five may be introduced in any order.

To minimize input, the user is allowed to omit cards two through six. All omitted values are assumed to be zero. The first element data card for an element must be card one and the last must be card six. Sort of cards two through five is immaterial.

No machine check is made between blocks of element data to insure data consistency. For an element, however, data is checked to make certain that a practical structure is being defined. This checking consists of insuring that quantities such as preintegrated geometry (such as thickness, moment of inertia, and cross sectional area), state identification (such as temperature and pressure levels), and surface temperature are positive entities and sufficient non zero data is included to define a structure.

The user may choose to introduce nodal coordinates in either a local or the overall coordinate system for a given element. All coordinates of the nodes of each element must be in the same system. Which coordinate system is being used is indicated by the data identity input on card 1. The option of defining element geometry simplifies input when a number of elements are bounded or lie on coordinate surfaces.

The user may sometimes choose, at each node, the coordinate system to which displacements are to be referred. This simplifies imposing boundary constraints since null rows are omitted from the stiffness matrix. Since this often results in fewer equations being handled, machine time is reduced as well.

The users option for the displacement coordinate system is limited in two ways. Firstly, all elements meeting at a common node must use the same displacement coordinates at that node. Thus, along a structural fold, the overall coordinate system will usually be required.

Secondly, the displacement coordinate system choice and data depend on the coordinate system used to define nodal coordinates. If the nodal coordinates are in the overall system, displacement components will be calculated in this system and a local reference system is not supplied. If, instead, the nodal coordinates are given in the local system, a local reference coordinate system is required. In this case displacement components may be calculated in either the local or overall system, depending on the signs used for the nodal numbers. The negative signed node numbers will have displacements referred to the local coordinate system. Note that in any event, the stresses are referred to a local coordinate system.

Nodal order is very important because it defines the positive normal to a plane. All adjacent elements must have the same positive direction. Because of the complexity of this topic, nodal order, the technical document should be consulted for a detailed explanation of this point.

Table 7-2 defines element data parameters and their fields on the six cards. On card one fields nine thru 12 are provided to handle geometries other than a facet. At present field eight is used in the FACET subroutine. On card two extra fields are also provided. On card five, data in fields 9-12 will always be ignored. The computer is directed to stop element data reading when card six is read.

Table 7-3 gives the interpretation of the parameters in Table 7-2. All items which depend on "Structural Type" are fully defined in the subroutine writeups under BILD subroutines. As stated before, FACET is the only BILD subreutine presently included.

Table 7-4 defines the computer data format for the parameters given in Table 7-3. This table provides all the necessary data for punching element input cards.

TABLE 7-2

ELEMENT INPUT DATA

Field 1		2	3	4	5	6	7	8	9	10	11	12
IA	IB											
Col 1	Cols 2-6	Cols 7-12	Cols 13-18	Cols 19-24	Cols 25-30	Cols 31-36	Cols 37-42	Cols 43-48	Cols 49-54	Cols 55-60	Cols 61-66	Cols 67-72
1	Element Number	Structural Type	Data Identity	Material Identity	Temperature Level	Pressure Level	Weight Level		Basic Geometric Data			
2			Surface Temperatures			and	Prestress Magnitudes					
3	Local Axis x_o	Local Axis y_o	Local Axis z_o	Local Axis x_a	Local Axis y_a	Local Axis z_a	Local Axis x_b	Local Axis y_b	Local Axis z_b			
4	4th Node Number	x_4	y_4	z_4	5th Node Number	x_5	y_5	z_5	6th Node Number	x_6	y_6	z_6
5	7th Node Number	x_7	y_7	z_7	8th Node Number	x_8	y_8	z_8	(No information goes in here)			
6	1st Node Number	x_1	y_1	z_1	2nd Node Number	x_2	y_2	z_2	3rd Node Number	x_3	y_3	z_3

TABLE 7-3

ELEMENT DATA INTERPRETATION

Card	Field	Interpretation
1	1A	Card Number, C: $1 \leq C \leq 6$
1	1B	Element Number, E: $1 \leq E \leq 999$
1	2	Structural Type:* A positive four digit word. The upper digit defines the number of nodes bounding the element and the lower two, the set of structural assumptions.*
1	3	Data Identity: A positive six digit word. The upper two digits define the coordinate system used for input; the middle two the temperature option; the lower two, the prestress option. Coordinates: >0, the overall rectangular cartesian system is used for nodes <0, a local reference system is being used for nodes. Temperature: 0, omit temperature stress and force calculations <0, include temperature stress and force calculations Prestress: 0, omit prestress calculations ≠0, include prestress calculations
1	4	Material Identity: A six character word identifying the material. This word must have a counterpart in the material tables.
1	5	Temperature Level: The temperature at which material properties are required.
1	6	Pressure Level: Normal pressure in the local plus z direction in pounds per inch or per square inch depending on structural type.*
1	7	Weight Level: >0, weight of the element; = 0, omit generation of weight vectors; <0, weight in pounds per inch, per square inch, or per cubic inch depending on structural type.*

*Refer to the specific technical document and BILD subroutine writeup.

Card	Field	Interpretation
1	8-12	Basic Geometric Data: Positive words. Interpretation depends on structural type.*
2	1B-9	Element State: Element temperature differences and initial stress values. The number of items of each type depends on the structural type.*
4-6	1B,5,9	Node Numbers; N: $-4095 \leq N \leq 4095$, $N \neq 0$. Negative numbers indicate use of the local coordinate system is desired at the node. The order in which nodal information appears on the cards determines the orientation of the local axes. The interpretation of the order depends on the structural type.*
4-6	2-4, 6-8 10-12	Nodal Coordinates: Coordinates at the nodes preceding the listing of coordinates. These coordinates will be in the overall system if the coordinate identity (Card 1, field 3) is zero and in the local system otherwise.

*Refer to the specific technical document and BILD subroutine writeup.

TABLE 7-4

FORMAT OF THE ELEMENT DATA PARAMETERS

Card	Col	Field	Format	Parameter	
1	1	1A	(I1)	Card Number	
1	2-6	1B	(2X,I3)	Element Number	
1	7-12	2	(2X,I4)	Structural Type	
1	13-14	3	(I2)	Coordinate System Indicator	Data Identity
1	15-16	3	(I2)	Temperature Indicator	
1	17-18	3	(I2)	Prestress Indicator	
1	19-24	4	(A6)	Material Identity	
1	25-30	5	(F6.0)	Temperature Level	
1	31-36	6	(F6.0)	Pressure Level	
1	37-42	7	(F6.0)	Weight Level	
1	43-72	8-12	(5F6.0)	Geometric Data	
2	1-54	IA-9	(I1, F5,0, 11F6.0)	Element Temperatures and Prestresses. Show specific format for facet here.	
3	1-54	1A-9	(I1, F5.0, 11F6.0)	Element Local Reference Place Location Data	
4-6	1-72	1A-12	(I1, F5.0, 11F6.0)	Element Node numbers and Nodal Coordinates	

SAMPLE MATERIAL TABLE INPUT

Field	1	2	3	4	5	6	7	8	9
	6061T6	535E+0	0.906E-6	16E+6	19E+6	41E+6	7E+6	3E+6	12E+7
-	72E+6	21E+6	37E+7	2E+6	11E+6	28E+6			

SAMPLE ELEMENT DATA

```
1    2    3001    000000    6061T6       535.0    40.0    1.0    0.25

6    2    D   0    0        3    0    1   0           4    1    0    0
```

7.4 Matrix Data

Matrix data format and interpretation are defined in Section 5. usage writeups in the operation subprogram READ. However, a few points of particular importance, such as formats and sort, are considered here.

There are two permissible formats, coded and precoded. If matrix data is not one of these formats a card reading stop will usually occur. For a detailed description of coded and precoded formats see Section 2-4.

Matrix data must conform with sort requirements. The identification card must be the first card. If the matrix is in coded format the sort must correspond to the sort designated on the identification card. The number of cards must be the same as defined on the identification card. If the matrix is in precoded format it must be in the order defined on the identification card and all zero terms must be included. The codes must appear first followed by the matrix elements.

7.5 Deck Arrangement

A problem deck is made up of three components and sometimes four. These components are: initiating program, pseudo instructions, description data, and sometimes matrix data.

The initiating program is an invariant. It is a self-contained preprogrammed package that starts the SAS program. It must be made a part of every new problem and it must be the first component in a problem deck.

The pseudo instructions constitute the second component of a problem deck. They must be made up according to the rules and formats outlined in Section 4. The last pseudo instruction must be a HALT.

Descriptive data is the third component. Descriptive data includes the "material tables" and the "element data" in that order. The material table must have at least three cards, two data cards and one zero stop card. The parameters of the cards and the associated formats are given in detail in Section 7.2. The element data follows the material table in the problem deck. The parameters and the formats are given in detail in Section 7.3 The technical document for the BILD subprogram being considered should also be consulted for additional pertinent details.

The fourth component which is optional is the matrix information. Matrix information consists of matrix data and heading cards. Matrix data is described in Section 7.4 and 5. The first matrix data card must be the identification card. The heading cards can contain any comments the analyst desires. They must be in the same order as they are to be printed. There is no practical limit to the number of cards.